1-1-2006

Generating background network traffic for network security testbeds

Hassan Javed Qureshi
Iowa State University

Generating background network traffic for network security testbeds

by

Hassan Javed Qureshi

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Co-majors: Computer Engineering; Information Assurance

Program of Study Committee:
Thomas E. Daniels, Major Professor
Steve F. Russell
Cliff Bergman

Iowa State University

Ames, Iowa

2006

Graduate College
Iowa State University

This is to certify that the master's thesis of

Hassan Javed Qureshi

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

DEDICATION

To my late grandparents – out of sight, but never out of mind

TABLE OF CONTENTS

LIST OF FIGURES ____ *vi*

ACKNOWLEDGMENTS ____ *vii*

ABSTRACT ____ *viii*

CHAPTER 1. INTRODUCTION ____ *1*

1.1 Multiple Modeling Paradigms ____ 3

CHAPTER 2. RELATED WORK ____ *6*

2.1 Network Security Specific ____ 6

2.2 Other Traffic Generation Tools ____ 8

2.3 Desirable Characteristics of Traffic Generator ____ 12

2.4 Analysis of Past Work ____ 12

CHAPTER 3. MOTIVATION ____ *14*

CHAPTER 4. BASIC ARCHITECTURE ____ *16*

CHAPTER 5. IN-DEPTH DESIGN ____ *18*

5.1 Data Acquisition and Statistical Processing ____ 18

5.2 Traffic Generation ____ 19

Algorithm ____ 27

CHAPTER 6. TRAFFIC GENERATION METHODS ____ *29*

6.1 WEB Traffic ____ 29

6.2 Interactive Traffic (Telnet, SSH, FTP and SFTP) ____ 32

6.3 Mail Traffic ____ 35

6.4 Side Effect Traffic ____ 36

CHAPTER 7. TOOLS USED ____ *38*

7.1 Network Analysis Tools ____ 38

7.2 Automation Tools ____ 39

CHAPTER 8. DATA VALIDATION AND VERIFICATION ____ *41*

8.1 Inter-session Time ____ 41

8.2 Number of Sessions ____ 44

8.3 Frequency of TCP Connections ____ 45

CHAPTER 9. FUTURE PLANS AND CONCLUSION ____ *46*

9.1 Future Plans ____ 46

9.2 Conclusion ____ 47

References ____ *48*

LIST OF FIGURES

Figure 1: Basic architecture of Markov Traffic Generator (MTG) 16

Figure 2: Main components of MTG 20

Figure 3: State User Behavior Model in a web session 31

Figure 4: Sample Telnet Session 33

Figure 5: Sample SFTP Session 35

Figure 7(a): Inter-session time between consecutive Telnet sessions from real traces 42

Figure 7(b): Inter-session time between consecutive Telnet sessions from simulated data. 42

Figure 8(a): Inter-session time between consecutive FTP sessions from real traces 43

Figure 8(b): Inter-session time between consecutive FTP sessions from emulated traces 43

Figure 10(a-b): Number of TCP connections per minute from real user and simulated user respectively 45

ACKNOWLEDGMENTS

I wish to express my profound gratitude to all those who made the successful completion of this journey possible.

Thanks to God - the most gracious, the most compassionate – for His love, guidance and countless blessings.

I would like to thank my family; especially my parents for their love, support and prayers. I thank my brother Bilal, my sisters Sobia and Reema for being so loving, cooperative and encouraging. I also like to pay gratitude to my loving grandparent (late), whom I owe everything I have.

Thanks to Dr. Tom Daniels for allowing me to work with him. I would like to acknowledge his patience, encouragement and assistance in providing thoughtful insights during all stages of the project. I really appreciate his support and guidance.

Finally, I thank all my friends and colleagues, both here and at home for being there whenever I needed their help.

ABSTRACT

With the advancement of science and technology, there has been a rapid growth in computer network attacks. Most of them are in the form of sophisticated and smart attacks, which are hard to trace. Although researchers have been working on this issue – attack detection, prevention and mitigation – the existing network security evaluation techniques lack effective experimental infrastructure and rigorous scientific methodologies for developing and testing the cyber security technologies.

To make progress in this area, we need to address one of the major shortcomings in evaluating network security mechanisms -- lack of relevant, representative network data. The research community is in need of tools that are able to generate scalable, tunable, and representative network traffic. Such tools are vital in a testbed environment, where they can be used to evaluate the behavior and performance of security related tools.

In this context, we present the Markov Traffic Generator (MTG), which is able to generate representative network traffic. The MTG follows a unique approach of generating background traffic at the session level, unlike the previous approaches operated on the packet level. The tool is application dependent and is able to generate various types of TCP traffic. The resulting tool is useful for researchers and developers in building, testing and evaluating cyber security related tools.

In this work, we develop the classifications of background traffic generation models based on the past work and present a new toolkit, the Markov Traffic Generator (MTG). As opposed to past work, MTG uses a first order hierarchical Markov agent to generate background user

behavior in network testbed. The Markov agents can be used to generate behavior that mimics observed traffic in real networks. The paper concludes by showing that MTG can realistically replicate observed network behavior

CHAPTER 1. INTRODUCTION

The Internet is growing so rapidly that it has become an important part of daily life. It provides one with the services like shopping, education, entertainment, all at the click of a button. With the convenience and luxury, it also brought in some complexities in ones life. These complexities are in the form of some type of computer crime. There has been an incredible growth in number of attacks occurring each year, which exponentially increased from six reported incidences in 1988 to 137,529 incidences in 2003 [25].

Over the past few years, there has been an increased investment in research on cyber security technologies by U.S. government agencies; even then we still lack technologies to protect a vital network. One reason for this deficiency is the lack of experimental infrastructures and rigorous methodologies for testing cyber security technologies. The network research community needs tools to evaluate new systems and algorithms that should be able to create a range of test conditions similar to those experienced in live network. This project addresses some of the shortcomings and presents a tool to help the research community to make the cyber world a secure and safer place.

As already discussed, computer users and researchers have been looking for tools and other ways to prevent cyber exploits. Two such tools for this purpose are firewalls and Intrusion Detection Systems (IDSs). An intrusion Detection System monitors network traffic for suspicious activity and takes appropriate action against the cyber criminal. As anomaly detection IDS identifies intrusions by differentiating the activity from normal activity on the network, therefore the description of *'normal traffic'* plays a vital role in the quality of

intrusion detection systems. The problem arises while testing these intrusion detection or intrusion prevention tools. While examining the effects and prevention methods of attacks, worms, and other problems computer criminals inflict on computers and networks, it is not practical to run experiments or projects on live network due to the fact that network could be damaged by the tests. The possible solution is to create a test bed network that would generate synthetic network traffic equivalent to the original network.

Simulating realistic network traffic takes into account various challenging factors that affect benign observed traffic in real networks. For example, the user behavior consists of usage habits as well as ordering of the applications that he uses; these characteristics make a user distinguished from other users. Similarly, number of application protocols, their types, their versions and protocol versions play an important role in behavior of generated traffic.

Along with the application and user level characteristics, there are parameters such as Time To Live (TTL) and Initial Sequence Number (ISN), which are operating system dependent. Therefore, variation of operating system may also contribute to the overall network traffic behavior.

The intra-protocol and inter-protocol behavior also affect the network traffic. The intra-protocol behavior consists of characteristics such as ordering, consistency, and semantics of application requests and responses. An example of this type of behavior is a FTP session, where a user logs in and uses commands '*ls*', '*get*', and '*quit*'. On the other hand, inter-protocol behavior is related to inter-dependence of protocols e.g., conversion from application requests to DNS and then DNS to ARP.

Apart from the above mentioned challenges, there are issues such as anonymization of payloads, spatial variations and temporal variations. Although the recent research has

addressed the privacy issue by using tools such as Tripp [35], work is still in progress to develop the automated anonymization tools. As traffic present in one network may differ from that in other networks depending on number of users, users' behavior, network location, day, time, and many other factors, the exact replication of network traffic is nothing less than a challenge.

1.1 Multiple Modeling Paradigms

Simulating background network traffic can be accomplished using several modeling paradigms. Conventionally, simulation has been done at the packet level, but the recent research such as Harpoon [36] has used flow level simulation for their traffic generation tool. However, there has not been any remarkable work done for traffic generation on session level.

Packet Level Emulation

In packet level emulation, network traffic is analyzed and emulated on a packet-by-packet basis. The basic characteristics of this type of emulation are packet inter-arrival times and packet sizes. One of the main advantages of this approach is simplicity. The parameters of packets are calculated independent of all other packets, which makes it less complicated. On the down side, these models ignore the fact that successive packets are not independent and the characteristics of one packet will often determine the characteristics of successive packets. This scheme is also inflicted with the problems of privacy in payloads, absence of inter-protocol and intra-protocol behavior, and temporal characteristics. Moreover, if packet-replay approach is used, then the generated traffic may not be clean or intrusion free.

Flow Level Emulation

Flow is defined as a single TCP connection that is started by the three-way handshake procedure and ended by the closing procedure. Flows do not start independently, they are generated by sessions. They are usually aggregated by source and destination IP addresses, source and destination port numbers and same input/output interfaces at router.

The advantages of this approach include a better way to model network traffic and generation of much smaller data sets as compared to the packet level measurements. On the down side, if flow-replay approach is used, this type of emulation is also imposed with privacy concerns in payloads because of the traffic anonymization requirement. Also, the traffic traces may already be corrupted, which is harmful for testing and training the IDS. On the other hand, if flow-replay approach is not used, then rewriting traffic for the client as well as for the server may be a problem. Moreover, absence of user behavior, operating system behavior and in some cases, applications behavior are some other issues concerned with this approach.

Session Level Emulation

In session level emulation, network traffic is analyzed at the session or conversation level. The analysis involves monitoring all connections on a network and recording information specific to each session. The relevant information about this connection would include session duration and session inter-arrival time. This model creates a number of unique sessions by using the information extracted from a small number of real network connections.

Although session level emulation adds complexity to the model, it also gives a great flexibility to manipulate the data collected from the real network that is being modeled. This

feature makes session level emulation much smarter than packet level emulation. It provides a method to control the output that a packet level model would be unable to match.

As there are many network applications, an accurate session level emulator needs to model all types of traffic. But for the purpose of most emulators, it is sufficient to model only the most dominant applications. Although modeling multiple applications increase accuracy, it also increases the computational overhead because of the rarely occurring network traffic. Therefore, for having a good balance between the accuracy of the tool and its efficiency, we need to generate dominant traffic only. For our current research purpose, we are concentrating on Web, Telnet, FTP, SFTP, SSH and Mail traffic.

MTG employs session level emulation to mimic the real behavior behind the interaction between users, protocols, and the network.

CHAPTER 2. RELATED WORK

2.1 Network Security Specific

This section discusses relevant previous work related to the testbed environment, which were specifically designed for network security research.

DARPA:

Several efforts have been made to develop a test bed for testing security tools. In 1998, DARPA initiated the Intrusion Detection Evaluation program. These evaluations involved generating background traffic combined with malicious payload that was collected in a period of seven weeks.

The DARPA test has been criticized for the following major reasons [34]:

- Statistics of the real traffic or the generated traffic were not published.
- Data rates and their variations with time was never a variable in the test environment.
- It is not known how many false alarms background traffic alone would generate.
- Inconsistencies in the documentation.
- Size of the training data may have been insufficient.

To address issues with DARPA, MIT Lincoln Labs developed a follow up testing environment called as Lincoln Adaptable Real-time Information Assurance Test-bed (LARIAT). The objective for creating LARIAT was to provide tools to assist in the evaluation and configuration of information assurance technologies. It gives flexibility to the user as one does not need to understand how the system works to set up the test bed for

evaluation. It "emulates the network traffic from a small organization connected to the Internet" [14]. Data collection and generation are not issues with this methodology as traffic scripts can be executed with different inputs to generate customized output specific to some network.

DETER/EMIST:

These projects were founded in 2004 with the major funding from NSF and US Department of Homeland Security. The DETER testbed is shared infrastructure designed for medium-scale experiments in computer security. Currently, the testbed has 231 nodes and it is being used for commercial and academic researchers to study attack [18]. It provides the ability to researchers to run experiments using risky code, on an isolated experimental network [18]. The Emulab controller, developed by the University of Utah, allows researchers to easily create network topologies with virtual LANS, as well as load operating systems, security systems, and other system parameters. Once loaded, the network is available for experimentation where actual machines can conduct security tests.

Another project that is closely related to DETER is Evaluation Methods for Internet Security Technology (EMIST). The mission of the EMIST project is the development of sound testing frameworks and methodologies for various forms of network attack and defense technologies. Within the EMIST tool suite, traffic generation is done with the help of the Scriptable Event System (SES) which is a tool that allows numerous hardware machines to be controlled from a central host. Execution commands are scripted and processed by a central machine, and are then passed down to subordinate machines where actual packets are dispatched. [21, 22].

In [23], the authors conducted DETER/EMIST DDoS experiments and observed obvious 'anomalies' due to the improper background traffic generation. Furthermore, they analyzed that mixing traffic being recorded in different network environment is not an easy task, especially for the purpose of anomaly detection system evaluation [23].

Our Markov Traffic Generation (MTG) tool considered the shortcoming from previous approaches and tried to mitigate them by following a unique application level network traffic generation, based on Markov model. The MTG tool is designed for computer-security experiments in medium-scale network such as, DETER and EMIST

RACOON

RACOON is a tool to generate user command data sets for evaluating user level anomaly detection systems. RACOON works on the assumption that a user's computational behavior is a causal process indicated by the commands he uses to accomplish a job or task [33]. They classified users into various classes such as programmers, scientists and system administrators, each with different job preferences and peculiarities in user command usage. In their model, commands are the first order description of user behavior, whereas, jobs considered to be the second order description.

Although RACOON is able to create variety of data sets to perform IDS evaluation, it cannot generate real data like other simulation tools [33].

2.2 Other Traffic Generation Tools

This section discusses other synthetic traffic generation tools, which are not specific to network security purpose.

TCPReplay and FlowReplay:

TCPReplay is a utility that takes very low level approach of replaying network flows. The tool is used to replay packets that were previously captured with the tcpdump program [4]. It takes a packet dump and replays each recorded packet without transport or upper layer knowledge. It also attempts to match the timing of the original traffic, either speeding it up or slowing it down [4].

Flowreplay is an enhancement of tcpreplay and is designed to replay traffic at Layer 4 (Transport Layer) or 7 (Application Layer) rather than at Layer 2 (Data Link Layer) like tcpreplay does. It has a goal of reading a pcap file, taking the client side of the connections and replaying that data using a standard TCP/UDP socket to connect to a server. This is because tcpreplay is unable to establish a valid TCP session with a server as it is unable to synchronize Seq/Ack numbers or maintain any TCP state. The major limitation of Flowreplay is that it is only capable of replaying the client side of a pcap against a real service on the target host [5].

Monkey:

Monkey is a tool to test the performance of web servers by varying the server behavior, network characteristics and client workload. It replays an emulated workload identical to the site's normal operating conditions [6]. It has two major components: Monkey See and Monkey Do. Monkey See is a tool for TCP tracing at a packet sniffer adjacent to the Web server being traced. Monkey Do is a tool for TCP replaying and consists of three emulators [6]. The client emulator replays client HTTP requests. The server emulator presents the

HTTP behavior of a Google server interacting with a client. Last, the network emulator recreates the network condition identical to the time the trace was captured.

Although Monkey is a good tool for testing the performance behavior of a web server, it has some limitations in cyber security environment. First, it cannot represent the real network traffic behavior as it is only able to generate Web traffic. Second, as the tool follows the record and replay approach, the emulated background traffic may not be absolutely clean.

Tomahawk:

Tomahawk is a tool for testing the performance and in-line blocking capabilities of IPS (Intrusion Prevention System) devices. It is a utility to replay tcpdump files bidirectionally at arbitrary speeds [7]. It is run on a machine with three network interface cards (NIC): one for management and two for testing. Two test NICs are typically connected through a switch, crossover, or NIPS. Tomahawk divides a packet trace into two parts [7]: The client packets, generated by the client, and the server packets, generated by the server 2. When Tomahawk replays the packet, the server packets are transmitted on eth1 and the client packets are transmitted on eth0 as default. If a packet is lost, the sender retries after a timeout period. If progress is not made after a specified number of retransmissions, the session is aborted. When the replay is finished, Tomahawk reports whether the replay completed or timed out. If the connection containing the attacks timed out, it implies that IPS blocked the attack successfully.

However, Tomahawk has some limitations [5]. First, it is impossible to have the generated traffic pass through routers because it can only operate across a layer 2 network. Second, it cannot handle traces containing badly fragmented traffic, and multiple sessions in the same

pcap can sometimes confuse it, Third, there is mishandling of TCP RST packet sent by an IPS.

SURGE [8]:

Surge is a tool to generate web workload. The tool is specially designed to examine the performance of servers and networks by varying the load. The idea behind this tool is to mimic a set of real users accessing a web server with the help of observations obtained from the real server. A user is modeled by an ON/OFF process (user equivalent), who during the ON period makes requests for Web files and during the OFF period lies idle.

SURGE (Scalable URL Reference Generator) generates references matching empirical measurements of 1) server file size distribution; 2) request size distribution; 3) relative file popularity; 4) embedded file references; 5) temporal locality of reference; and 6) idle periods of individual users [8]. Since SURGE is only able to generate Web traffic, it is not suitable for Information Assurance environment.

Representative Traffic for IDS Benchmarking [9]:

Kayacik and Heywood [9] proposed a framework for generating data for training and testing of intrusion detection systems. The tool can develop models of web usage from web server logs in a data driven fashion and the actual traffic is generated based on those logs on the host. In their work, they employed Markov model for transitional probabilities between requests and transitional delays [9]. In our tool, we also used the Markov Model to capture temporal properties such as, transitional probabilities between different traffic sessions and inter-arrival time between sessions. This way this model is closely related to our traffic

generation tool. The limitation for this tool is that it is only capable of generating the web traffic, which is not sufficient for thorough testing of intrusion detection systems.

2.3 Desirable Characteristics of Traffic Generator

Here are some of the desirable characteristics for a traffic generation tool:

1- It must be able to generate realistic network traffic i.e., traffic that closely mimics human behavior on Internet.

2- It should be able to generate heterogeneous traffic from wide variety of protocols.

3- It should be straightforward to make any modification in the model if e.g., number of sources would be changed.

4- It should be able to repeat the same experiment several times with same starting parameters.

5- It should be able to scale large number of hosts.

6- It must be able to run continually with minimal human supervision.

7- It should be able to generate traffic on the basis of statistics given for different time intervals.

8- For IDS benchmarking, the traffic generation tool should generate attack-free data.

9- The tool should have a capability to extract the statistical characteristics from observations of real networks.

2.4 Analysis of Past Work

From our desirable characteristics mentioned above, we analyzed various popular traffic generation tools in market today and realized the lack of necessary characteristics in a single synthetic traffic generation tool. For example, some of the above mentioned tools are

application specific and can only generate a single type of traffic i.e., web traffic in most of the cases. These tools include Monkey [6], SURGE [8] and the tool presented in [9]. Some of these tools were specially designed for the purpose of load-testing on routers, servers and bridges. Therefore they are not suitable for Information Assurance environment.

Research on network workload generation has typically focused on creating models based on packet level or flow level. Tools like TCPReplay and FlowReplay follow the same idea for replaying the network flow. The problem with this approach is that the recorded traffic traces may already contain malicious payloads. These traces may ruin the performance of intrusion detection or prevention systems and result in the form of undetected or un-prevented attacks. This approach also cannot scale well on a high speed network and therefore, is not considered effective for evaluation of IDS.

In summary, most of the previous network traffic generation tools have some special features that distinguish them from other tools. Although these tools work for a specific purpose such as traffic generation for a web server or load testing for routers, switches and firewalls, they cannot be recognized as all-purpose traffic generator as numbers of them fail to perform well in information security environment.

CHAPTER 3. MOTIVATION

Our motivation in this research is based on a focus to design a scalable internet traffic generator that can be easily used specifically for the purpose of generating background traffic in network security testbed. In our design, we tried to fulfill most of the requirements mentioned earlier under desirable characteristics section. Our MTG tool has the following main characteristics:

- Our main goal is to simulate normal user behavior using different network protocols. The current prototype is particularly focused on mocking the user's behavior in a real network.
- MTG is capable of generating many traffic scenarios depending on different testbed configurations. It is designed to support major TCP traffic at layer 7 (application layer), such as, HTTP, Telnet, FTP, SFTP, SSH and Mail
- Unlike most of the tools, it is able to repeat an experiment by using the same seed values.
- Because testing of anomaly detection system is dependent on normal behavior, which is learned from the training data, clean traffic plays an important role in this regard. MTG does not replay the already recorded sessions, it is able to generate intrusion-free background traffic.
- The tool is scalable and can work with many hosts without degrading the overall performance.
- The tool is easily configurable and relies on statistical models given by the user. These statistical models are also easy to modify by editing configuration files.

- Finally, the network traffic is constantly changing over time both in volume and statistical properties, even if it observed at the same location. Therefore, provided the right statistical models from different time intervals, the tool is able to generate the representative synthetic traffic.

We believe that with the diverse and increasing number of network traffic, a traffic generation tool that requires few statistical parameters to achieve its objective is imperative. Moreover, we created an approach that is different from other approaches as it simulates artificial traffic on the basis of users and sessions, instead of packets or flows and hence can be used to simulate user behavior in testbeds.

CHAPTER 4. BASIC ARCHITECTURE

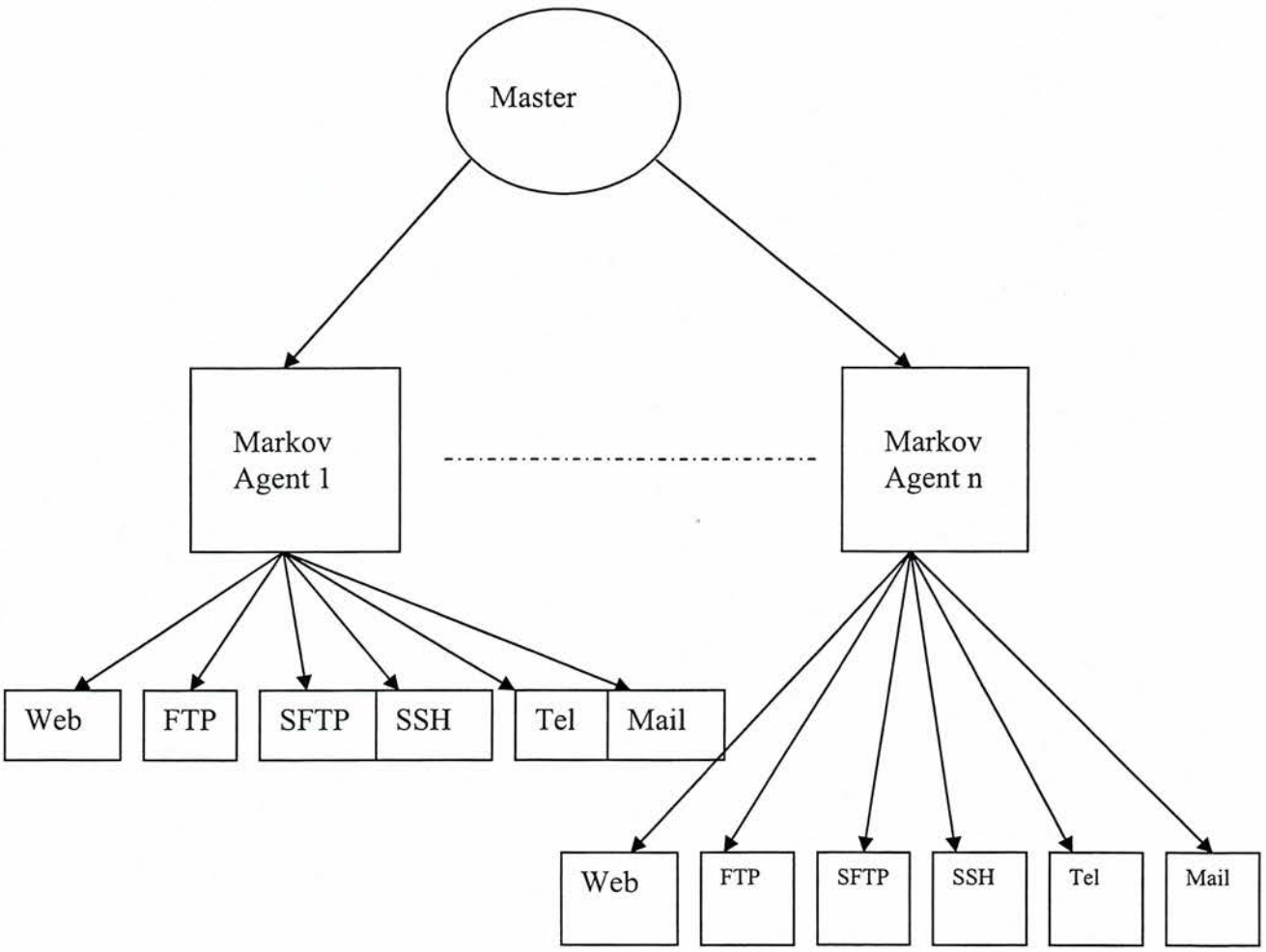


Figure 1: Basic architecture of Markov Traffic Generator (MTG)

The MTG uses a distinct modeling approach by incorporating many Markov agents to generate actual background traffic in a testbed. The agents implement a hierarchical Markov model of order 1. The states represent sessions, whereas transition represents moving from one session to another session. We use a master to many slave architecture to coordinate the agents.

Master: Master node is the controlling host of our tool. It interacts with number of hosts and is considered to be responsible for doing these tasks:

- Passing configuration files from user to different hosts for running Markov agents. This configuration file contains parameters and statistical information extracted from the real network traffic.
- Running time for each Markov Agent.

Markov Agent: It is considered to be the brain of this tool and is responsible to generate synthetic traffic based on the following characteristics from the real network traffic.

- Running time of different sessions.
- Probability for next session after each specific session type.
- Session inter-arrival times.

CHAPTER 5. IN-DEPTH DESIGN

Our designing approach has the following main components:

1. Data Acquisition and Statistical Processing:

   Acquire the data in raw format and extract statistical attributes out of it to represent the traffic.

2. Traffic Generation:

   Generate the synthetic traffic based on the attributes found in the previous step

3. Data Verification:

   Comparing and correlating the artificial traffic with the real network traffic.

Next, each component will be described in detail.

5.1 Data Acquisition and Statistical Processing

The initial phase is to collect data for the network that user wants to model. This can be achieved by using any of the network monitoring programs such as tcpdump. Tcpdump is the utility to collect the incoming packets as well as outgoing packets over a specified network and then store it in a trace file [12]. Useful statistics can be extracted from these raw files that can be later used by Traffic Generation module to generate a similar type of traffic.

Next step is to process the trace files to obtain useful parameters out of it. As our MTG tool is an application dependent tool, which requires the session level statistical parameters, the task is not as trivial as it is in case of packet level. The major problem is that there are no tools available that can give us the required information on a higher level such as, session or

application inter-arrival time. Due to this reason, we developed our own Perl scripts that take out the required parameters from input files. Our scripts are able to extract the following major types of parameters:

- Session inter-arrival times
- Application frequencies
- User's duration
- Application's average duration

All of these parameters help to model a specific type of user on the basis of type of applications he uses, time to spend on different applications, user's life etc.

Other than our own scripts, flow level analysis tools such as Softflowd[1] and Nfdump[10] are useful in obtaining the information on packet level or on flow level.

5.2 Traffic Generation

Our architecture model is different from other models by introducing the concept of users as Markov Agents. Each user type has associated applications to use and each of these applications has its own set of commands to execute. The model is hierarchical i.e., master node is controlling all the hosts that exist in the network. The user needs to give configuration to the master node that distributes a configuration file among other hosts. Figure 2 shows the architecture of the mechanism.

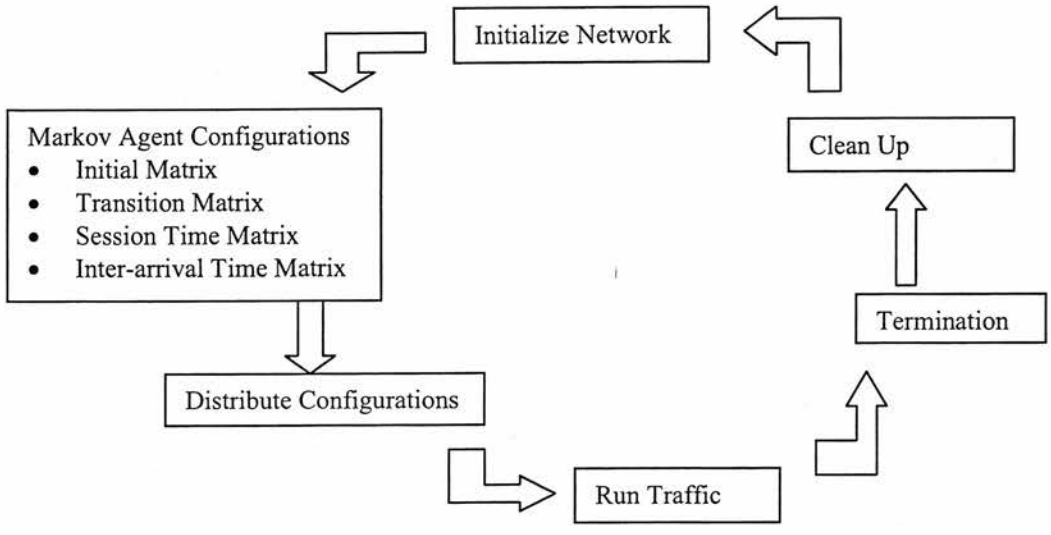


Figure 2: Main components of MTG

- **Initialize Network**: The first step user needs to do is to configure the traffic profile. This includes number of users of each type that need to be simulated and the total time duration to run the traffic. For example, if a user wants to simulate a small network for half an hour consisting of ten workstations, in which three of them are being used by administrators, two are used by researchers and remaining five are used by students, then he needs to put in the following parameters in the configuration file:

    1. Total number of users = 10

       Administrators = 3

       Researchers = 2

       Students = 5

    2. Time duration = 30 min

Master computer has a whole list of workstations connected to it. It will randomly pick up the workstations equal to total number of users and let each of them work as an

independent user for the specified time. All threads are stopped once the time duration is over, resulting in giving back the control to the Master node.

- **Markov Agents:** In our model, markov chains are used as a statistical tool to emulate each type of user's behavior. This way each station will act as a markov agent that exhibits the same properties as that of a real user from some category defined by a set of parameters.

Definition 1: Markov Chain is a finite state machine with probabilities for each transition, that is, a probability that the next state is S_j given that the current state is S_i [30].

Therefore Markov Chain is a stochastic process in which the future development depends only on the present state, but not on the past history of the process or the manner in which the present state was reached.

$$P(X_t = j \mid X_0 = i_0, X_1 = i_1, \ldots, X_{t-1} = i_{t-1}) = P(X_t = j \mid X_{t-1} = i_{t-1})$$

If the state of the system at time *t* is dependent only on the state of the system at time *t-1* plus some independent random component, the process is referred to as *first-order Markov Chain.* In a *second-order Markov Chain* the state of the system at time *t* would depend on the states of the system at both *t-1* and *t-2*.

In MTG, state is referred to session of some particular type of traffic. On the other hand, transition is conversion from one session to another session. For example, if the tool is generating *Web* session, its current state is considered to be *'web'*; a conditional probability to have a particular next session given that the current session is *Web* can be referred as transitional probability.

For creating a Markov Agent, we need to have the following four types of matrices for each type of user:

1. ***Initial Matrix***: This matrix represents the initial statistics for all the available type of traffic sources. For example, if a user is student, then he may have the following matrix:

$$\textit{Initial Matrix} = I_{I=} \begin{pmatrix} \text{Web} = P_W \\ \text{FTP} = P_F \\ \text{SFTP} = P_{SF} \\ \text{SSH} = P_S \\ \text{Telnet} = P_T \\ \text{Mail} = P_M \end{pmatrix}$$

This vector shows that there is P_w probability of generating the web traffic as initial traffic source. The default values would be set for different categories of users, but one has an option to modify them before running software.

2. ***Session Time***: This matrix represents the average session time of each type of traffic. The mean time and the standard deviation are used to model the session duration from the real network traces. MTG tool is configured on these statistical parameters to simulate the session durations, which are in close approximation with the real traffics' session durations.

$$\textit{Session Time Matrix} = \begin{pmatrix} \text{Web} = T_W \\ \text{FTP} = T_F \\ \text{SFTP} = T_{SF} \\ \text{SSH} = T_S \\ \text{Telnet} = T_T \\ \text{Mail} = T_M \end{pmatrix}$$

T_i is the session time parameters for traffic type *i*, where *i* = {Web, FTP, SFTP, SSH, Telnet, Mail}

3. ***Transition Matrix*:** The transition probabilities, the P_{ij} which give the probability that the process will move from state S_i to state S_j are given for every pair of states. It is a matrix of probabilities to go from one traffic session to another. Therefore, it is a square matrix where columns and rows, both represent different types of traffic. In some of the cases, diagonal entries usually have more probabilities as compared to other entries i.e., simulated user generate the same type of traffic as that of last session.

$$\textit{Transition Matrix} = T_{ij} = \begin{array}{c} \\ \textbf{\textit{WEB}} \\ \textbf{\textit{FTP}} \\ \textbf{\textit{SFTP}} \\ \textbf{\textit{TELNET}} \\ \textbf{\textit{SSH}} \\ \textbf{\textit{MAIL}} \end{array} \begin{array}{c} \begin{array}{cccccc} \textbf{\textit{WEB}} & \textbf{\textit{FTP}} & \textbf{\textit{SFTP}} & \textbf{\textit{TELNET}} & \textbf{\textit{SSH}} & \textbf{\textit{MAIL}} \end{array} \\ \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} & P_{15} & P_{16} \\ P_{21} & P_{22} & P_{23} & P_{24} & P_{25} & P_{26} \\ P_{31} & P_{32} & P_{33} & P_{34} & P_{35} & P_{36} \\ P_{41} & P_{42} & P_{43} & P_{44} & P_{45} & P_{46} \\ P_{51} & P_{52} & P_{53} & P_{54} & P_{55} & P_{56} \\ P_{61} & P_{62} & P_{63} & P_{64} & P_{65} & P_{66} \end{bmatrix} \end{array}$$

Attributes on the left represent current states, whereas, those over the matrix represent the destination states corresponding to each of the given current state. Therefore the i^{th} row of the transition matrix is the probability distribution of moving to all states from state *i*. Given the training data, probability of transition from state *i* to state *j* can be calculated as follows:

$$P_{ij} = \frac{A(S_t = j, S_{t-1} = i)}{\sum_i A(S_{t-1} = i)}$$

Where the numerator is the number of times state *j* follows state *i* in the training data.

4. ***Inter-arrival Time Matrix***: This matrix represents the inter-arrival time distribution (time between two consecutive sessions). There are various distribution functions available that could be used for different types of transitions e.g., Poisson distribution, exponential distribution, normal distribution and uniform distribution. Each of these distribution functions require different parameters, which include mean(μ), standard deviation(σ), minimum outcome (*min*) and maximum outcome (*max*). Methods to calculate each of the available random number generators are discussed below:

   Uniform Random Number: If *r* is a random number from a standard uniform distribution [27], then

   $$x = a+(b-a)r$$

   is a random number from a uniform distribution with the range from *a* to *b*.

   Normal Random Number: To transform a standard normal variable *z* into a normal random variable *x* with mean μ and standard deviation σ, the following relationship [27] is used:

   $$x = \mu+ \sigma z$$

   Poisson Random Number: Generation of random numbers from a poisson distribution with a mean of λ can be accomplished by multiplying successively generated standard uniform random numbers [27]. Specifically, multiply *N* standard uniform random numbers U_i until

$$\prod_{i=1}^{N} U_i < e^{-\lambda}$$

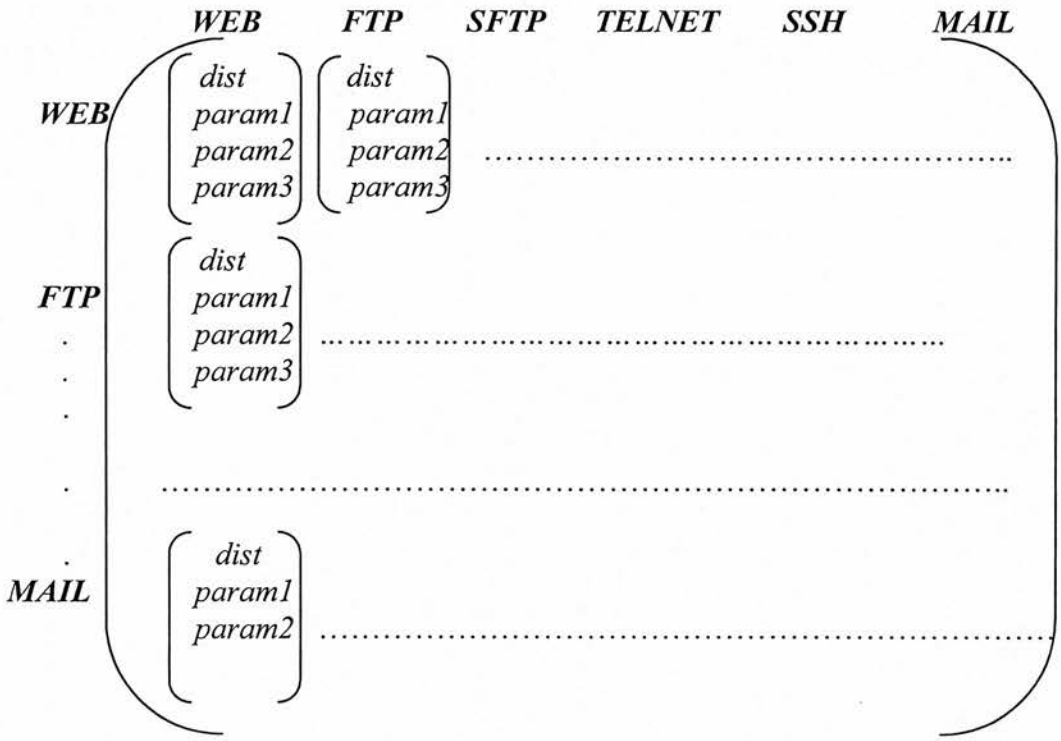
Then the value of the poisson random variable X is $N-1$.

***Exponential Random Number*:** The generation of a uniform random number r permits the formation of an exponentially distributed random number by [27]:

$$X = -1/\alpha \ ln\ (r)$$

Where α represents the mean.

The user has the flexibility to select a distribution function that closely represents his network environment. Default parameters are already set, but user can easily edit them in the configuration file. Also the tool has a flexibility to add new random number distribution functions with minimal changes in one of the scripts.



Each of the entity in the above matrix consists of four parameters. First one is the distribution type. As each of these distributions needs different set of parameters

for execution e.g., uniform distribution needs number of outcomes, minimum and maximum, whereas, normal distribution needs number of outcomes, mean and standard deviation; *param1, param2* and *param3* represent the three parameters corresponding to each of the distribution. Some of the distributions like Poisson need only two parameters i.e., number of outcomes and mean.

- **Distribute Configuration:** The traffic configuration and simulation details are distributed to all the hosts. Each of the host will act as an independent user and configure the parameters corresponding to its user type i.e., if a simulated user type is "student", then the host will have access to the matrices meant for students.

- **Run Traffic:** Once the configuration file has been distributed, its time to run traffic from each of the configured hosts. Master workstation is responsible for triggering a host to generate a specific user's type traffic. Therefore, Master node will pass two arguments to other hosts; 1) user type to mock 2) total time to run.

- **Termination:** After simulation time is over, control is given back to the Master computer from each of the hosts. Continuing user sessions are killed to ensure that no more traffic is being generated.

- **Clean Up:** It changes the configuration file and markov agents back to the default settings. It takes to the Initialization phase to make sure that all experiments start from a common state, even if a traffic session is terminated before completing.

Algorithm

Input: User type, Time to run, Initial Matrices for each user type, Session Time Matrix for each user type, Transition Matrices for each user type, Inter-session Matrix for each user type

Output: Statistical Equivalent Traffic

Begin:

I <- usertype.initial_matrix;

S <- usertype.session_time_matrix;

T <- usertype.transition_matrix;

D <- usertype.intersession_matrix;

While (*time1*) // where variable 'time1' is the time given by master node

ctr <- 0;

If *ctr* = 0 // To be executed for the first time

session <- ***Stat_Fcn*** (I);

time2 <- ***Session_Time*** (session);

repeat until (*time2*)

execute *session*;

last_session <- *session*;

ctr = *ctr* +1;

end

else // Repeat until 'time1' duration ends

session <- ***Transition*** (*last_session*, T);

```
                time2 <- Session_Time (session);
                repeat until (time2)
                        intersession_delay <- Delay (last_session, session, D);
                        sleep (intersession_delay);
                        execute session;
                        last_session <- session;
                        ctr = ctr +1;
                end
        end
end
```

The *Stat_Fcn* function takes an initial matrix as input, selects one of the session on the basis of probability and output the resulting session. The *Session_Time* function tales a current session type as an input and outputs a probable session time The *Transition function* takes two arguments as input, the last session and the transition matrix of a specific user type. This function chooses a most probable session on the basis of probabilities given in the transition matrix. The *Delay function* calculates the inter-session delay time between two consecutive sessions.

CHAPTER 6. TRAFFIC GENERATION METHODS

Currently, we have six different types of traffic to be generated synthetically. These include interactive applications such as Telnet, SSH, FTP and SFTP, as well as non-interactive applications such as Web traffic and Mail. New modules can be added for other types of traffic.

6.1 WEB Traffic

World Wide Web traffic compromises of major portion of the total traffic on the internet. A recent measurement study by Sprint [26] on 19 of the OC-48 links in their network found that on 16 of them, the largest application class was web traffic that ranged from 31% to 59% of the total bytes transferred. On the other 2 links, web traffic was the second largest with 16% to 31% of the total bytes. According to [28], web traffic modeling is difficult for two reasons.

1- Many of the system components interact with one another. Web browsers and Web servers from different vendors behave differently and have different parameter values.

2- A web interaction becomes more complex because of the changing nature of the Web environment. A user may open multiple browsers and generate pages from these browsers at the same time.

Because of the above mentioned problems, it is hard to say that there is any single template of web interaction. Therefore, we tried to design a model for web traffic on general basis that is able to be applied to real time traffic most of the times. Also, as our tool is user based, therefore there are different probabilities of web traffic for each type of user.

Session in Web Traffic

Packets and flows can be easily identified on the real traces. A flow is a single TCP connection that starts with the three-way handshake procedure and ends with the closing procedure. Sessions, on the other hand, are well-defined in case of interactive traffic such as Telnet and FTP, but it is more difficult to provide a specific and unique definition of a session for HTTP traffic. This could be because of the absence of log-in or log-out behavior such as the one in interactive sessions. There could be many different definition proposed on the basis of traces; for example, all the web pages downloaded from the same web server in a limited period of time can form a session.

Recent research has developed various models to define web session. In [31], He et al. determined session boundaries by examining the inter-arrival times of URL requests. The assumption is that URL requests that belong to the same session will have short inter-arrival times, whereas long inter-arrival times between requests will mark the end of session. In our architecture, we define the web session as "the web session starts when a user opens some website, browse its children links, and lasts until it visits another website that is different from the previous one." In the definition, children links mean those links that we can *wget* from some webpage.

User Behavior In Web Session:

2 states model will be used to describe the user behavior in a web session. Those states are defined as follows:

1- Download: User opens a connection to a URL-address, selected from a list of addresses based on probabilities, downloads the webpage, parses through it, and downloads all corresponding image files and applets.

2- Read: User reads the downloaded webpage and considers what to do next, download another one, or close the web session?

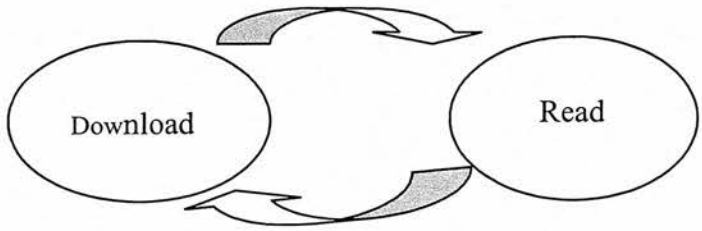


Figure 3: State User Behavior Model in a web session

Each user who starts to download a web page enters the download communication state. When the web page and all its contents are downloaded, the user is directed to the Read state. The Web module describes a web browser that downloads web pages from real web servers. The browser downloads the source file of the web page, parses this, and downloads the images and java applets identified on this page. These are also downloaded with the download of the web source file.

Two URL-address lists will be used, parent URL-list and children URL-list. The parent URL-list contains addresses of popular sites from all over the world. This list can easily be changed. The tool has a flexibility to choose URL-addresses either randomly or on the basis of probability i.e., some sites are assigned higher probability then others and therefore have greater chance to open. Same web site cannot be visited consecutively, as that does not depict a common user behavior. Also, some bad requests (web sites unavailable) would be entertained to make this model close to a real web traffic model.

The other list is children URL-list. This list is dynamically updated as new pages are visited. This is done by inclusion of all the HREF addresses when parsing through the source file of a web page. The URL-address from this list is selected randomly and does not depend on any probability function.

Web Generation module will select a URL from either the parent URL-list or children URL-list. The probability of choosing the children URL-list is sufficiently higher then selecting parent URL-list. This is because in real web traffic, user tends to visit children web pages more as compared to new web sites.

The web generation module gives user a lot of flexibility to generate traffic of its own choice. User can change the URL-list to download web pages of his choice. Similarly, he has an option to increase the probability of some sites as compared to other sites. There is no bound on least or most number of addresses that means he can include as many URL-addresses as he wants. User can also modify the probability function to select one of the URL-list more times or equal times as other list.

6.2 Interactive Traffic (Telnet, SSH, FTP and SFTP)

Interactive traffic plays an important role in overall traffic generation. Currently we have all the major types of interactive traffic modules available that include Telnet, SSH, FTP and SFTP.

Each traffic session has its own commands that mostly do not work with other traffic sessions. Therefore, each distinct application would contain the list of commands that it can execute. Our tool could either grab these commands in random sense or on the basis of probability i.e., some commands have more probability to be executed as compared to other.

Telnet and SSH

"Telnet" and "SSH" are types of terminal session (also called "terminal emulation") programs that allow users to conduct interactive sessions with computers elsewhere on the Internet by providing a character-based (i.e., non-graphical) connection. Although they have almost the same functionalities, SSH is considered to be more secure and reliable. A Telnet or SSH session starts when a user logs into some host through one of these protocols and lasts till the user logs out.

The tool uses the functionalities of Expect, which is a program to talk to other interactive programs according to a script. To begin telnet and SSH sessions, the respective Expect scripts need to be generated and then triggered. For example, for generating a Telnet session, Expect will generate a script having different commands related to Telnet following the user name and password. This 'Telnet.exp' script needs to be called by a program to get executed for traffic generation.

```
set timeout -1
spawn $env(SHELL)
match_max 100000
send -- "telnet 192.168.24.4\r"
set send_human {.4 .4 .2 .5 5}
expect "login:"
send -h “user\r"
expect "Password:"
send -h “pass\r"
```

Figure 4: Sample Telnet Session

Short Telnet and SSH sessions with impermissible usernames or password were introduced to mimic the behavior of someone who forgot his password or someone who is trying to break in the network.

FTP and SFTP:

FTP and SFTP are two different types of clients that provide the same service of file transfer. FTP is a standard protocol for transferring files to and from remote machines running FTP servers. The SFTP client uses the SSH (Secure Shell) File Transfer Protocol to provide secure file transfer over a non-secure network. Both the protocols allow file and directory upload, download, and directory navigation of remote and local file systems. Expect is used to automate the file transfer connection between client and server. Multiple user accounts are created in the server, so that it would not give impression that only one user is generating all the traffic. Also, tool user has an option to configure the tool in such a way that each user group has different access privileges assigned to it. This makes it closer to real traffic model where some users can only access the files but cannot delete them. The definition of session is self explanatory in case of FTP and SFTP sessions, as a session starts when a user logs into the host machine through one of these protocols and lasts till he logs out following the transfer of files to or from host server.

In order to make the traffic more realistic, we produced short sessions with impermissible logins and passwords. The characteristic depicts the behavior from some user who has forgotten his password or someone who want to break in, like in a real network environment.

```
set timeout -1

spawn $env(SHELL)

match_max 100000

expect "\$"

send -- "sftp user@192.168.22.4\r "

expect "Password:"

set send_human {.4 .4 .2 .5 5}

send -h "pass\r "

expect "sftp> "
```

Figure 5: Sample SFTP Session

6.3 Mail Traffic

Mail traffic is also known as SMTP traffic and is considered to be the second most well-known traffic seen in TCP connection. SMTP is the de facto standard for e-mail transmission across the internet [13]. It is relatively simple text-based protocol for transferring the text messages after specifying recipients of a message. It uses TCP port 25 and is a "push" protocol that does not allow one to "pull" messages from a remote server.

For generating SMTP traffic in the testbed, we had to setup a DNS server and Mail server. DNS server is a system that stores information associated with domain names in a distributed database on networks [13]. In this case, it is used to accept emails for some specific domain and redirect those emails to the mail server. MX record or Mail Exchange record is a record in DNS that is used to point to the mail servers to send the email to, and which one it should

be sent to first on the basis of priority. On the other hand, Mail Transfer Agent (MTA) is used to implement SMTP. They are responsible for routing and delivery of email. In our testbed environment, we used BIND as our DNS server and POSTFIX as an MTA. Both of the servers are Linux based.

We had two choices for recipients of the generated email traffic. First, send the traffic to the real recipients on outside world, who include the users on popular email websites such as, hotmail or yahoo. Second method was to create multiple user accounts on different machines in the testbed and generate the email traffic internally in between the hosts. We ended up in selecting the second approach as it is better for the research and testing purpose.

Three different lists are used for generating the synthetic email traffic.

1. Sender Address: This list contains the addresses of the senders in specific host.
2. Recipient Address: It contains the addresses of all the recipients in all the hosts in testbed.
3. Attachment Files: It contains the file names, which are available to be attached with any outgoing email. These files vary in their sizes.

These parameters are enough to produce statistical similar email traffic as that of the real network. Furthermore, the email traffic is generated by using a Perl script, which is easy to modify and configure according to the user's requirements.

6.4 Side Effect Traffic

Some TCP and UDP traffic such as, Domain Name Services (DNS) queries and Network Time Protocol (NTP) is automatically generated along with the application traffic. DNS uses

a port 25 and is a set of formalized rules that explains how data is communicated over a network. NTP, on the other hand, uses port 123 and provides time synch between computers and network systems.

CHAPTER 7. TOOLS USED

Here are the tools that we used as components in building the Markov traffic generator:

7.1 Network Analysis Tools

TCPDump:

Tcpdump is one of the most popular tools to capture and dump the traffic on a network [12]. It prints out the headers of packets on a network interface that match the Boolean expression. The output of tcpdump is protocol dependent. The data stored in tcpdump format can be analyzed by majority of the network analysis tools usch as, Ethereal and SoftFlowd [1].

Flow Tools:

Flow-tools is a library and a collection of programs used to collect, send, process, and generate reports from NetFlow data. The tools can be used together on a single server or distributed to multiple servers for large deployments [2]. We used it for the purpose of collecting network flows.

Flows are exported from a router in a number of different configurable versions. A flow is a collection of key fields and additional data. The flow key is {srcaddr, dstaddr, input, output, srcport, dstport, prot, ToS}. Flow-tools supports one export version per file [2].

SoftFlowd:

It is a flow-based network traffic analyzer capable of data export. Softflowd semi-statefully tracks traffic flows recorded by listening on a network interface or by reading a packet capture file [1]. This tool is used on the traffic traces captured by flow tools.

NFDump and NFSen [10]:

Nfdump is a packet level and flow level analysis tool that process netflow data on the command line. It consists of the following built-in tools [10]:

- nfcapd – netflow capture daemon
- nfdump- netflow dump
- nfprofile - netflow profiler
- nfreplay- netflow replay
- nfclean.pl- cleanup old data
- ft2nfdump- read and convert flow-tools data

For our research purpose, we used two of the above mentioned tools. First one is ft2nfdump, which is used to convert the data collected from flow-tools to netflow. Second tool is nfdump, which is used to analyze the flows by using various filters such as, source/destination IP address, source/destination ports, protocols, number of bytes, number of packets etc. Moreover, the syntax of nfdump is similar to tcpdump.

Nfsen is a graphical tool, which uses nfdump as a backend tool [11]. It can show the graphs ranging from the current network situation to details down to the specific flow.

7.2 Automation Tools

Expect:

Expect is a tool for automating interactive applications such as Telnet, FTP, passwd, fsck, rlogin, tip, etc [3]. Expect really makes this stuff trivial and is also useful for testing the same

applications. By adding Tk, we can also wrap interactive applications in X11 GUIs [3]. We used this tool to automate Telnet, FTP, SFTP and SSH sessions.

Expect was designed specifically to interact with *interactive* programs. An expect programmer can write a script describing the dialogue. Then the expect program can run the "interactive" program non-interactively. It can also be used to automate parts of a dialogue, as control can be passed from the script to the keyboard and vice versa. The language of Expect is based on Tcl, which is a subroutine library that becomes embedded into an application and provides language support.

CHAPTER 8. DATA VALIDATION AND VERIFICATION

In this section we will compare results of traffic generated by our tool with the original traffic from the network.

Although there are many data traces available on internet, there are no tools to extract the statistical parameters on session level. Most of the currently available tools either work on packet level or on flow level such as tcpdump[12] and Ethereal[16]. Due to the unavailability of the tools, we had to write our own Perl scripts to extract the parameter related to our requirements. Furthermore, we evaluate our results using Lincoln lab datasets [29] as benchmark. All of the datasets that we used were consisted of attack-free traffic.

8.1 Inter-session Time

Inter-session, inter-flow or inter-packets times play important role in emulating the real network traffic. Because our MTG tool is based on session level, we will only consider the inter-session time in real network traffic. By using those parameters from inter-session time into out MTG tool, we will validate that our tool generates the traffic with the same statistics as that of the real network traffic.

Telnet Inter-session Time

For our first experiment, we recorded the inter-session time between consecutive Telnet sessions in Lincoln lab datasets. We observed that inter-session time depicted the characteristics from *Exponential* distribution. This behavior is shown in Figure-7(a). By configuring our MTG tool with the collected parameters, we generated the traffic that has

inter-session time closely related to the real network traffic. Figure-7(b) shows the result in the histogram form. These plots clearly show that the intersession time in between the sessions of the same type i.e., Telnet in this case, have the same distribution.

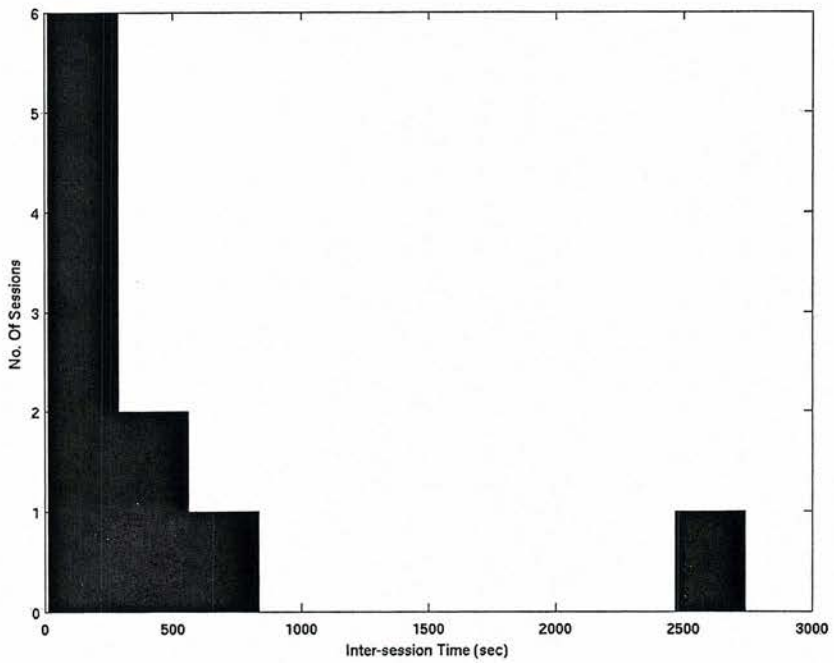


Figure 7(a): Inter-session time between consecutive Telnet sessions from real traces.

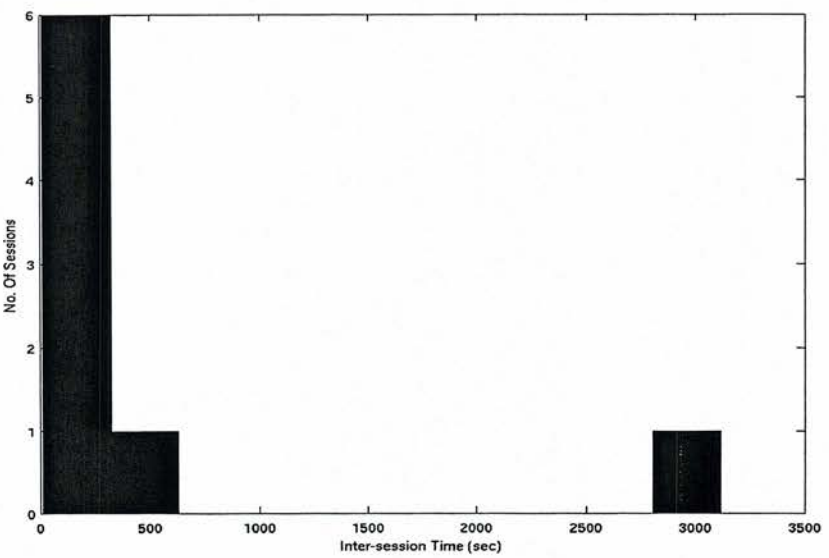


Figure 7(b): Inter-session time between consecutive Telnet sessions from simulated data

FTP Inter-session Time

With the help of Lincoln lab datasets, we collected the inter-session time for consecutive FTP sessions. The behavior observed is similar to the one in Telnet Inter-session time. Figure-8(a) shows the results from the real network traffic, whereas, Figure-8(b) shows the results from synthetic traffic generated by our MTG tool.

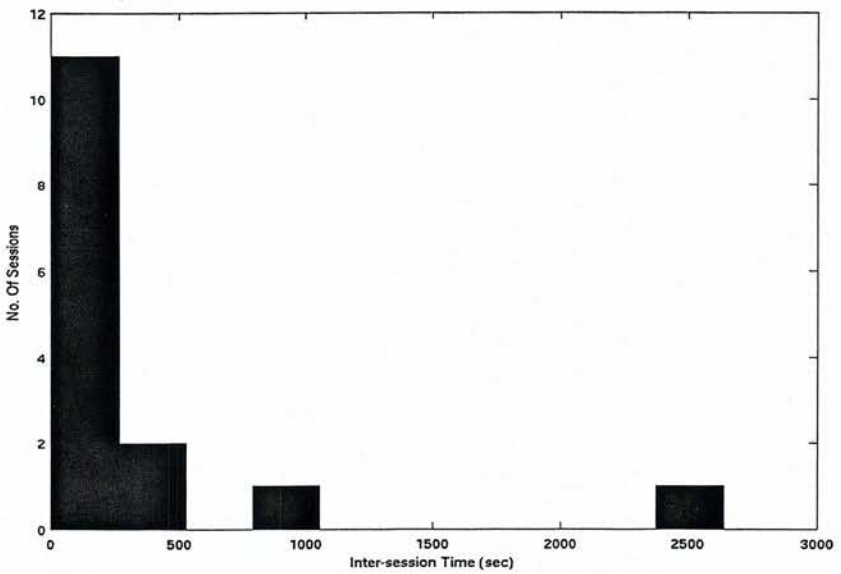


Figure 8(a): Inter-session time between consecutive FTP sessions from real traces

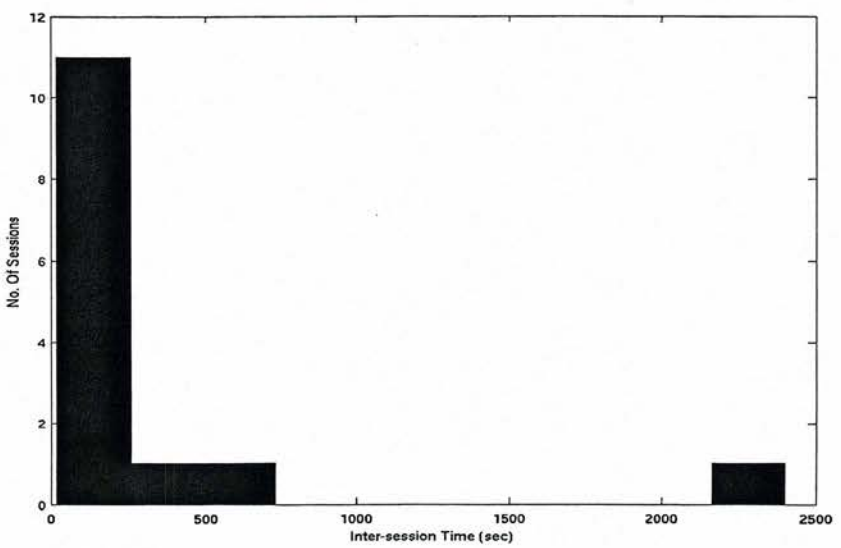


Figure 8(b): Inter-session time between consecutive FTP sessions from emulated traces

8.2 Number of Sessions

For validating the user level model, we picked one of the users from Lincoln labs network traces. We observed that the user generated three types of traffic namely, Web traffic, SMTP traffic and FTP traffic. Figure-9(a) shows the statistics for number of traffic sessions of different protocols from real traces. Figure-9(b) shows the number of sessions generated by an emulated user from MTG tool.

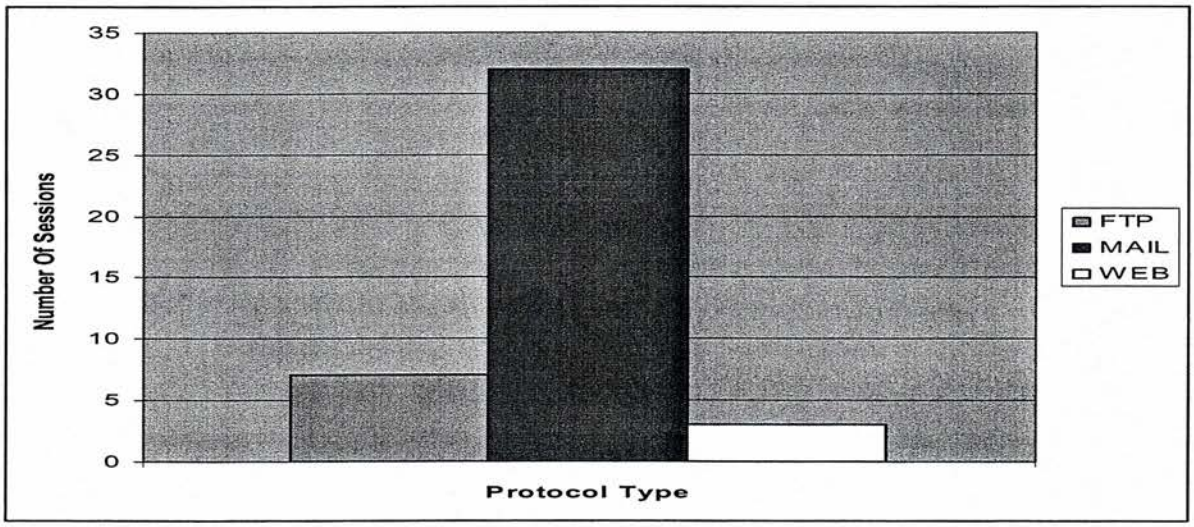


Figure 9(a): Number of session from a real user

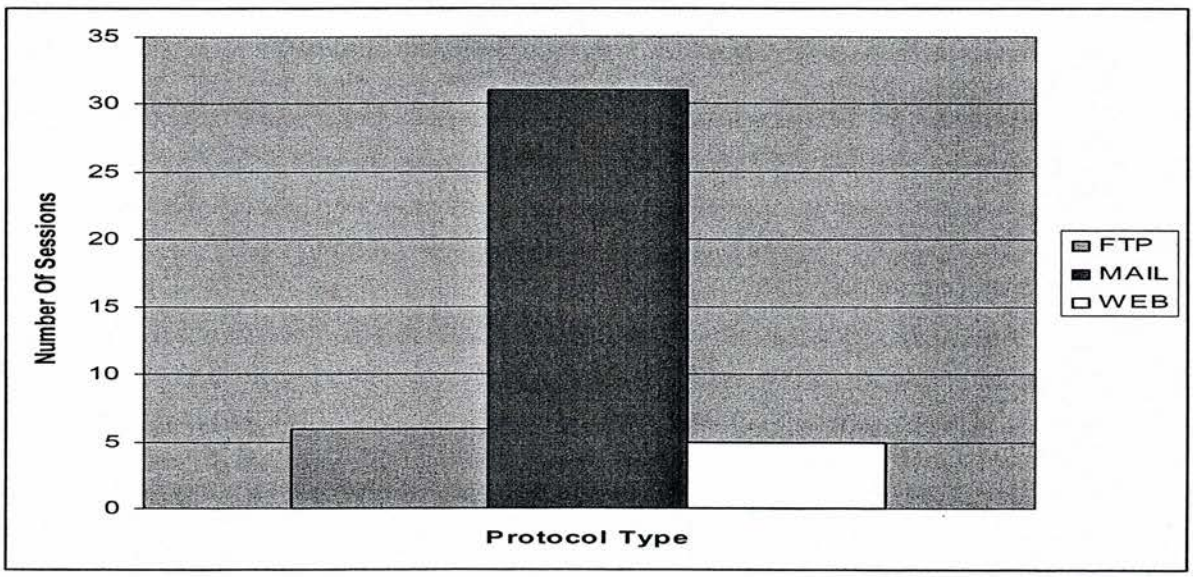


Figure 9(b): Number of session generated by emulated user via MTG tool

8.3 Frequency of TCP Connections

TCP connections make the major part of our network traffic, as our tool is configured to launch only TCP connections with the current settings. We chose one of the users from Lincoln labs datasets and simulated his behavior with our MTG tool. The real user generated FTP and SMTP applications frequently; therefore the emulated traffic was also mostly consisted of these two protocols. Figure-10(a) and 10(b) shows the TCP connections per minute for the real user and simulated user respectively. The result shows that the distribution of TCP flows per minute in the emulated traffic is consistent with the distribution in the real traffic.

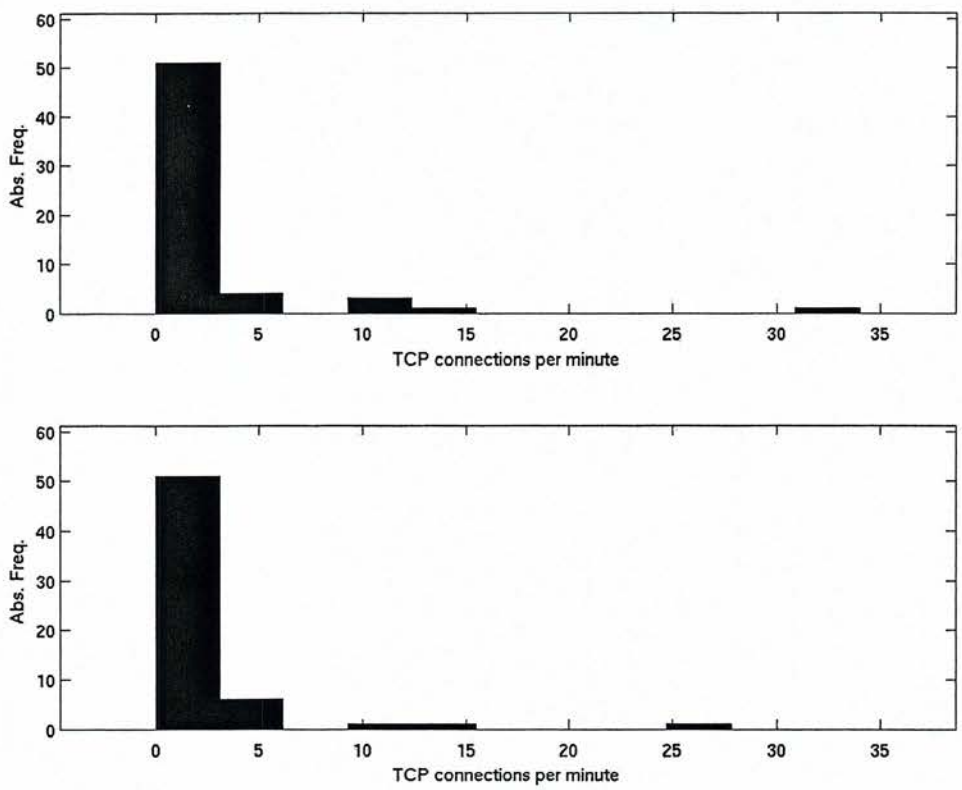


Figure 10(a-b): Number of TCP connections per minute from real user and simulated user respectively

CHAPTER 9. FUTURE PLANS AND CONCLUSION

9.1 Future Plans

- Although the MTG tool lets user to give his specification for traffic generation, but still we need a default model that could represent a real traffic behavior in any good sized network. For this purpose, we need to obtain statistics of different traffic from network administrators of some large based networks. As traffic behavior in small network will be different from statistics of large network, therefore we are planning to provide user different models on the basis of the size of network. Then user has an option to select a default model that best represents his test-bed.
- Currently, we have six different types of network services available in MTG tool. Most of these services are provided by TCP. We need to inject more traffic models that use UDP protocol. Applications like multimedia, chat and DNS use UDP protocol for delivery of their packets. By adding some popular UDP traffic, we can make this tool to generate traffic that would be really close to the real network traffic.
- To make the traffic generation and result presentation more user friendly, a Graphical User Interface (GUI) would be an easy development of the Implementation. .
- For the extraction of statistical parameters from real traces, we are using number of third-party tools. These tools may have different syntax that require user to get familiar with. It would be nice if there is a feature in MTG tool that can extract the required statistics automatically, given the real network trace in the tcpdump format.
- Xen[32] or Xen World[37] could be used for simulating the large scale deployment of

MTG tool. Xen is an open source virtual machine monitor, which is developed by the University of Cambridge. Its design goal was to run 100 full-featured operating system instances on a single typical computer [13].

By incorporating MTG tool with Xen Worlds, we are able to simulate hundreds of users by using couple of actual machines. The high performance virtualization makes the MTG tool really useful, not only for the purpose of network security testing but also for load testing.

9.2 Conclusion

Our goal for MTG was to create an efficient, statistically accurate and highly customizable synthetic network traffic generator that is useful for researchers and developers in the field of Information Security. The MTG tool operates at Session level and simulates the users at the same level; this means that it tries to mimic the Application level or Session level characteristics from real network traces such as, session inter-arrival time, session transition probabilities and number of specific type of applications. The tool also incorporates many statistical functions that help it to mimic the real network traces accurately. We believe that the tool will be accepted as a benchmark, especially for testing in the Information Security environment.

We have shown that the artificial traffic produced by MTG is close to the actual network traffic. Our results also show that the tool is able to model the network, both in terms of individual characteristics and overall pattern. Moreover, the MTG tool provides accuracy with flexibility at a different level than the more traditional traffic generators.

References

[1] Softflowd- flow-based network traffic analyzer.

http://www.mindrot.org/softflowd.html (date retrieved: March 4, 2006).

[2] Flow-tools- tool set for working with Netflow tools

http://www.splintered.net/sw/flow-tools/docs/flow-tools.html (date retrieved: March 5, 2006)

[3] Expect – tool for automating interactive applications.

http://expect.nist.gov/ (date retrieved: January 15, 2005).

[4] TCPReplay - traffic replay utility.

http://tcpreplay.sourceforge.net/ (date retrieved: February 20, 2006)

[5] S. Hong, S. F. Wu. *"On interactive internet traffic replay"*. In proc. of RAID Seattle, WA, USA. 2005: 247-264.

[6] Yu. Cheng, U. Holzle, N. Cardwell, S. Savage, G.M.Voelker. "*Monkey See, Monkey Do: A Tool for TCP Tracing and Replaying"*. In proc. of USENIX technical conference 2004.

[7] Tomahawk- a performance testing tool.

http://www.tomahawktesttool.org/tutorial.html (date retrieved: February 23, 2006)

[8] P. Barford and M. Crovella. *"Generating representative workloads for network and server performance evaluation*". In proceedings of ACM SIGMETRICS 1998, pages 151-160, Madison, WI, June 1998.

[9] H. Gunes Kayacik, Nur Zincir-Heywood. *"Generating representative traffic for intrusion detection system benchmarking"*. In proc. of IEEE Communication Networks and Services Research Conference, 2005, pages 112-117, May 2005.

[10] Nfdump – tool for collecting and processing netflow data.

http://nfdump.sourceforge.net/ (date retrieved: March 5, 2006).

[11] P. Haag. "*Watch your flows with NfSen and Nfdump*" http://www.ripe.net/ripe/meetings/ripe-50/presentations/ripe50-plenary-tue-nfsen-nfdump.pdf (date retrieved: March 5, 2006)

[12] Tcpdump - tool for collecting and analyzing network data. http://www.tcpdump.org/ (date retrieved: February 10, 2006)

[13] Wikipedia – encyclopedia http://en.wikipedia.org/wiki/ (date retrieved: March 23, 2006).

[14] Rossey, Lee M.; J. C. Rabek.; R.K. Cunningham; D.J. Fried.; R.P. Lippmann and M.A. Zissman. *"LARIAT: Lincoln Adaptive Real-time Information Assurance Testbed"*, In proc. of IEEE Aerospace conference, 2002. Vol.6, 2002 Pages: 6-2671-2676, 6-2678-6-2682 vol.6.

[15] Tcpreplay – tool for replaying the captured network traffic. http://tcpreplay.sourceforge.net/. (date retrieved: August 2004)

[16] Ethereal – network traffic analyzer www.ethereal.com (date retrieved: February 5, 2006)

[17] Marcus J Ranum., *"Experiences Benchmarking Intrusion detection Systems"*, www.nfr.com, May 5, 2002.

[18] Terry Benzel, R. Braden, D. Kim, C. Neuman, A. Joseph and K. Sklower. *"Experience with DETER:A testbed for security research"*. In proc. of 2nd international IEEE conference on testbeds and research infrastructure for the development of networks and communities. March 1-3, 2006.

[19] Emulab- the total network testbed. www.emulab.net (date retrieved: March 15, 2006).

[20] Deterlab – the network security testbed based on Emulab.

www.deterlab.net (date retrieved: March 15, 2006).

[21] EMIST – Evaluation Methods for Internet Security Technology

http://emist.ist.psu.edu (date retrieved: March 15, 2006).

[22] Scriptable Event System (SES from EMIST)

http://www.cs.purdue.edu/homes/fahmy/software/emist/documentation.html

(date retrieved: March 15, 2006).

[23] S. Hong, F. Wong, S. Felix Wu et. al. "*TCPtransform: property-oriented TCP traffic transformation*". Detection of Iintrusion and Malware & Vulnerability Assessment (DIMVA) conference. Wien, 2005.

[24] N. Athanasiades, R. Abler, J. Levine, H. Owen, G. Riley. *"Intrusion Detection Testing and Benchmarking Methodologies"*. In proc. of first IEEE international workshop on information assurance, March 24, 2003. Pages: 63-72

[25] CERT Coordination Center. Statistics

www.cert.org (date retrieved: April 06, 2006)

[26] S. Bhattacharyya. C. Diol. R. Gas, E. mess, S. Mmn. A. Nucci, D. Papagiannaki. and T. Ye. *"Packet Trace Analysis*," Sprint Labs, Tech. Rep.. 2003,

ipmon.sprintlabs.com.

[27] Udo W. Pooch, James A. Wall "*Discrete Event Simulation – A Practical Approach*", CRC Press, ISBN 0-8493-7174-0

[28] H.K Choi, J.O. Limb. *"A behavioral model of web traffic*". In IEEE proc. of ICNP 1999, Pages 327-334.

[29] Lincoln lab datasets

http://www.ll.mit.edu/IST/ideval/data/data_index.html (data retrieved: February 12, 2006)

[30] National institute of standards and technology (NIST)

http://www.nist.gov/dads/HTML/markovchain.html (data retrieved: March 12, 2006)

[31] D. He, A. Goker. "Detecting session boundaries from web user logs". Proceedings of the 22nd annual colloquium on information retrieval research (ECIR), April 2000

[32] Xen-virtual machine monitor

http://www.cl.cam.ac.uk/Research/SRG/netos/xen/ (date retrieved: March 25,2006)

[33] R. Chinchani, A. Muthukrishnan, M. Chandrasekaran, S. Upadhyaya. "RACOON: Rapid generating user command data for anomaly detection from cuztomizable templates". Proceedings of 20th annual computer security application conference, 2004.

[34] J. McHugh. "Testing intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln laboratory". ACM transactions on information and system security, vol. 3, no. 4, November 2000,pp.262-294.

[35] TRIPP Traffic Rewriter for IP (v4) Packets.

`http://tripp.dynalias.org/` (March 17, 2006)

[36] J. Sommers, P. Barford. Self-configuring network traffic generation. Proceedings of the 4th ACM SIGCOMM conference on Internet measurement, Taormina, Slicily, Italy, 2004.

[37] B. Anderson, Thomas E. Daniels. Xen Worlds: Xen and the Art of Computer Engineering Education. In proceedings of 2006 ASEE Annual Conference and Exposition, June 2006.